

▶ ПЕРФОРМАНСНА АНАЛИЗА НА KEY-VALUE БАЗИ НА ПОДАТОЦИ: СПОРЕДБА НА REDIS И POSTGRESQL

Изработка: Ангела Иванова 211104, Бојан Ристов 211151,
Данче Јованова 211156, Моника Габрешанец 211091

Ментор: Проф. д-р Слободан Калајџиски

Вовед





- ▶ Key-Value базите се едни од најприменуваните, со едноставна архитектура заснована на парови клуч-вредност. Тие нудат висока брзина, лесна употреба и хоризонтална скалабилност, што ги прави погодни за апликации со високи перформансни барања.
- ▶ Овој проект има за цел компаративна анализа на перформансите на Redis - како native Key-Value база - и PostgreSQL - како релациона со Key-Value поддршка. Анализата ќе се изврши врз реален датасет со филмски метаподатоци од IMDB и TMDb со над еден милион записи и повеќе од четириесет атрибути, со цел да се утврдат предностите, ограничувањата и препораките за нивна практична примена.

Методологија

- ▶ Проектната задача се реализира преку практична имплементација на Key-Value модели во две различни бази на податоци: Redis (како native NoSQL решение) и PostgreSQL (како релациона база со NoSQL проширувања).
- ▶ Методологијата опфаќа неколку чекори:
 1. Инсталација и конфигурација на базите на податоци
 2. Моделирање на податоците
 3. Импортирање на податоците
 4. Избор и извршување на сценарија за пристап до податоците
 5. Перформансна анализа

Инсталација и конфигурација на PostgreSQL

- ▶ PostgreSQL беше инсталиран преку Docker Compose, со што се овозможува брзо подигање, изолација и лесна репродукција на околината.
- ▶ Конфигурацијата е дефинирана во `docker-compose.yml`
- ▶ **Моделирање на податоците**- За PostgreSQL креиравме една табела (movies табела)
- ▶ **Импортирање на податоците** - Импортирањето се реализира преку Python скриптата `02_import_data.py`, која извршува неколку клучни задачи: Прочистување на податоците, Мапирање на колони, Батч-импортирање

Name	Container ID	Image	Port(s)	CPU (%)	Memory usage...	Memory (%)	Stack read/writes	Network	Actions
postgres	45c4b6f661c	postgres:15-alpine	5432:5432	0.00%	284.0MB / 15.2%	3.83%	0B / 0B	3.0KB/s	 
pgadmin	c4e68c3ca22	dpage/pgadmin4:latest	8080:80	0.00%	261.7MB / 7.64%	3.34%	0B / 0B	1.42KB/s	 

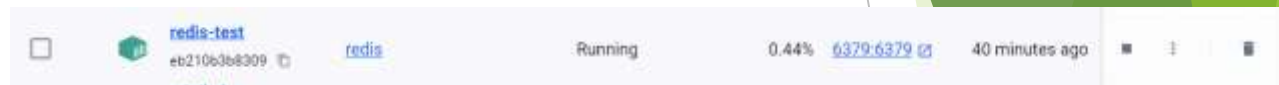
```
services:
  postgres:
    image: postgres:15-alpine
    container_name: movie_postgres
    environment:
      POSTGRES_DB: movie_db
      POSTGRES_USER: user
      POSTGRES_PASSWORD: password
      POSTGRES_HOST_AUTH_METHOD: trust
    ports:
      - "5432:5432"
    volumes:
      - postgres_data:/var/lib/postgresql/data
      - ./scripts:/scripts
    restart: unless-stopped

  pgadmin:
    image: dpage/pgadmin4:latest
    container_name: movie_pgadmin
    environment:
      PGADMIN_DEFAULT_EMAIL: admin@admin.com
      PGADMIN_DEFAULT_PASSWORD: admin123
    ports:
      - "8080:80"
    depends_on:
      - postgres
    restart: unless-stopped

volumes:
  postgres_data:
```

Инсталација и конфигурација на Redis

- ▶ За потребите на проектот, Redis е инсталиран и конфигуриран локално користејќи Docker Desktop, што овозможува брза и изолирана средина за работа со базата на податоци.
- ▶ Redis инстанцата е подигната со командата:
docker run -name redis -p 6379:6379 -d redis
- ▶ Поврзувањето со Redis се реализира преку Python библиотеката **redis-py**, користејќи ги стандардните параметри localhost и порта 6379.
- ▶ **Моделирање на податоците** - Податоците се складираат во Redis со key-value пристап, каде секој филм има клуч во формат movie:. Дополнително, се користат сетови и сортирани збирки за организирање и индексирање.
- ▶ **Импортирање на податоците** - Импортирањето во Redis се реализира со Python скрипта која ги вчитува првите 5000 филмските записи од CSV датотеката и ги зачувува како JSON стрингови со формат на клуч movie:<id>.



```
1 import redis
2 import json
3
4 #loading dataset
5 df = pd.read_csv(io.BytesIO(b"IMDB TMOB Movie Metadata Big Dataset (1M).csv"), low_memory=False, nrows=5000)
6
7 df = df.dropna(subset=["id", "title"])
8
9 redis_db = redis.Redis(host='localhost', port=6379, db=0)
10
11 for _, row in df.iterrows():
12     key = f"movie:{int(row['id'])}"
13     value = json.dumps(row.dropna().to_dict())
14     redis_db.set(key, value)
15
16
17 print("Data loaded into Redis.")
```

Прашалници

За тестирање на функционалностите и перформансите на базите на податоци PostgreSQL и Redis, беа дефинирани и извршени вкупно 9 прашалници, поделени во три категории според сложеноста на операциите:

► Едноставни прашалници

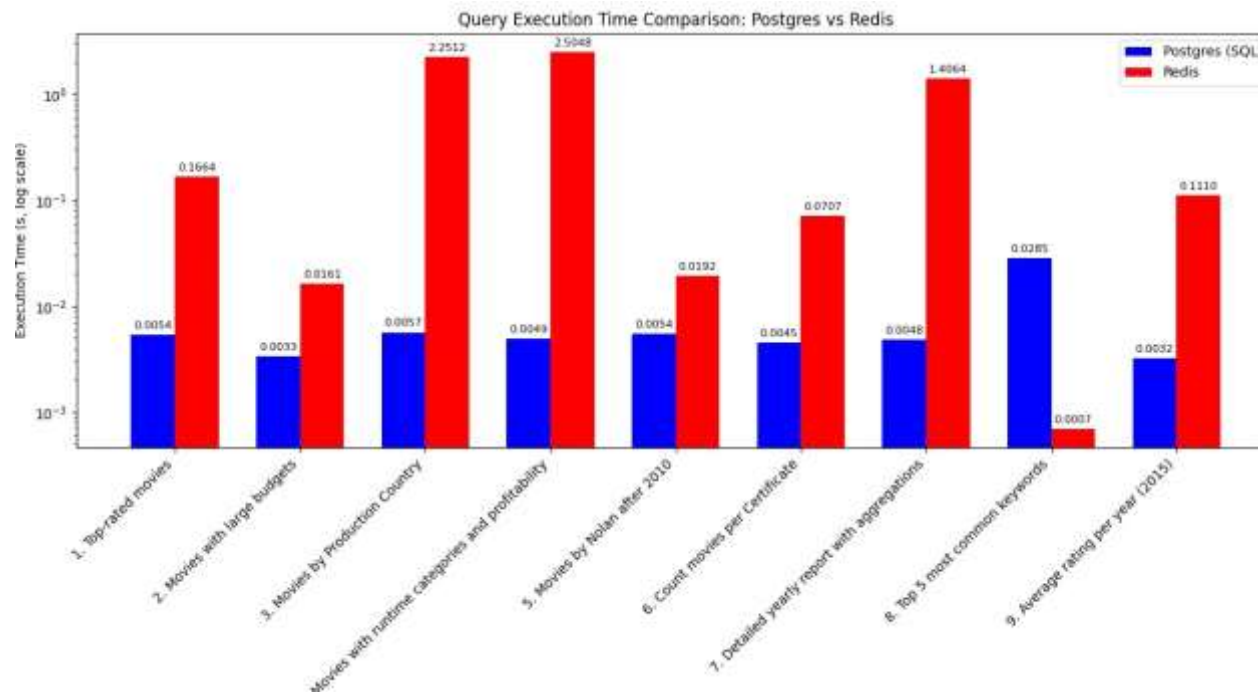
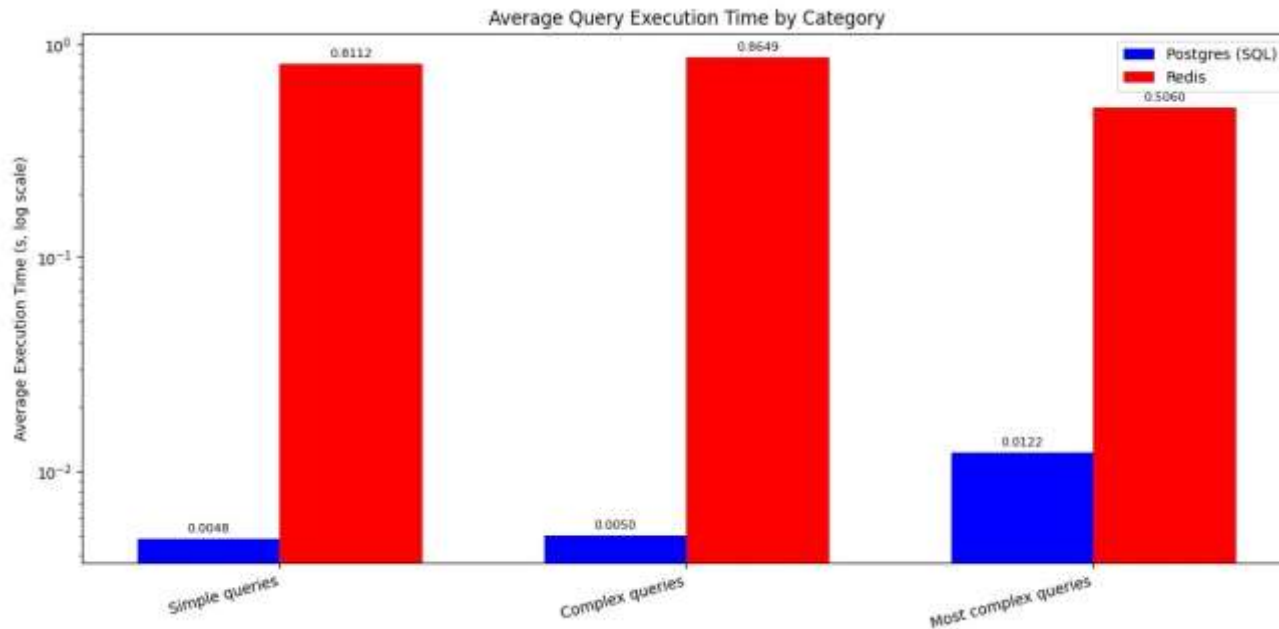
- Највисоко оценети филмови (Top-rated movies) - филмови со рејтинг поголем од 8.0.
- Филмови со голем буџет (Movies with large budgets) - филмови со буџет над 100 милиони.
- Филмови по држава на продукција (Query movies by Production Country)

► Сложени прашалници

- Анализа на runtime категории (Movies with runtime categories and profitability) - филмови категоризирани по должина (долги, кратки, стандардни) и нивната профитабилност.
- Movies by Nolan after 2010 - филмови на режисерот Кристофер Нолан по 2010 година.
- Финансиски пресметки (Count movies per Certificate) - број на филмови по категоризација според сертификат.

► Многу сложени прашалници

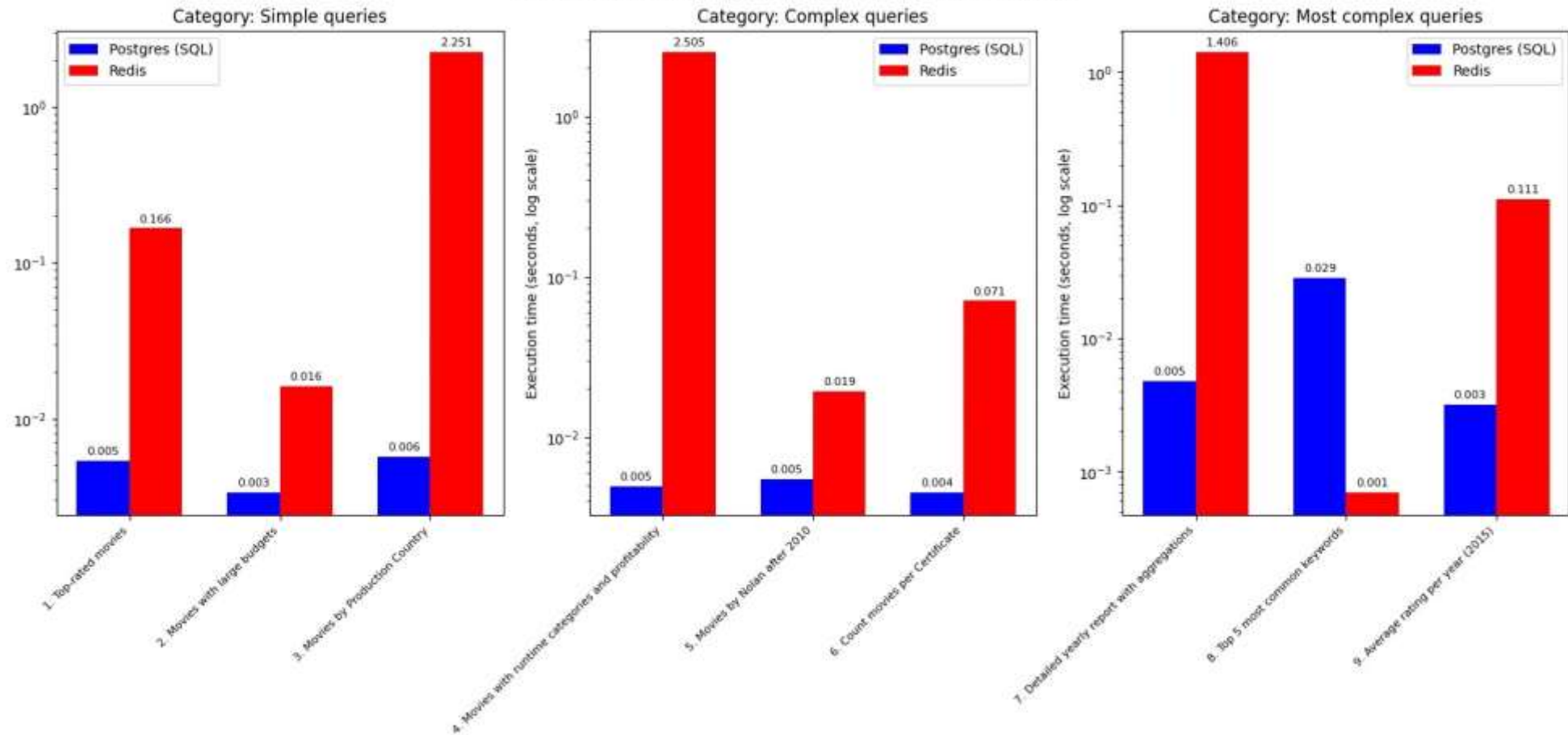
- Детален годишен извештај - број на филмови, просечен рејтинг и финансиски показатели по година.
- Top 5 most common keywords - најчесто користени клучни зборови во филмовите.
- Average rating per year - просечен рејтинг по година.



Перформансна анализа

- ▶ **Едноставни прашалници:**
PostgreSQL има просечно време на извршување од околу 0.005s, додека Redis во истите случаи е значително побавен со просек од околу 0.81s.
- ▶ **Сложени прашалници:**
PostgreSQL повторно покажува стабилни резултати со просек од околу 0.005s, додека кај Redis просекот се движи околу 0.86s.
- ▶ **Многу сложени прашалници:**
Кај PostgreSQL времето расте, но останува релативно ниско со просек од 0.012s, додека Redis има просечно време од околу 0.51s.

Comparison of query execution times by category



Заклучок

- ▶ Резултатите од мерењата покажаа дека PostgreSQL е значително побрз при сите категории на прашалници - едноставни, сложени и многу сложени. Времињата на извршување се движат во опсег од 0.005 до 0.012 секунди, што укажува на висока оптимизација на релациониот модел за аналитички и агрегатни операции.
- ▶ Од друга страна, Redis, иако е познат по својата in-memory архитектура и ефикасност кај едноставни key-value операции, покажа значително подолги времиња на извршување за комплексни прашалници. Ова укажува дека Redis е попогоден за кеширање и брз пристап до поединечни податоци, додека PostgreSQL е супериорен за аналитички и извештајни задачи.

Ви благодариме
за вниманието!